NOM Prénom :

CSC3102 – Contrôle Final 1
Année 2023 – 2024

Consignes :

- répondez aux questions en écrivant soit dans les cadres fournis au fil du sujet, soit dans les listings des scripts placés en fin d'exercice, aux endroits des marques correspondant aux questions ;
- si vous manquez de place, vous pouvez écrire sur la page 5 **en l'indiquant dans l'encart initial ;**
- les réponses qui ne requièrent pas de code doivent être justifiées ;
- polycopié, notes personnelles manuscrites et exemples de script sont autorisés ;
- la durée est fixée à 1h30 ;
- le barème est donné à titre indicatif.

Billetterie des olympiades d'informatique

Victimes de leur succès, les organisateurs des olympiades d'informatique se voient obligés de réguler l'accès à leur billetterie en mettant en place un sas d'attente. Pour acheter des billets, un client -- dont le comportement est simulé par le script *client.sh* -- fait une demande d'accès auprès d'un guichet unique qui joue le rôle de sas d'attente, modélisé par le script *sas_attente.sh.* Ce sas est doté d'un nombre d'accès aux serveurs de la billetterie de l'évènement qu'il attribue aux clients demandeurs dans la limite des stocks disponibles (dans cet exercice, le réapprovisionnement en entrée du sas n'est pas traité). Un client ayant ainsi obtenu un droit d'accès à la billetterie est alors placé en attente d'un serveur de billetterie libre. Si les accès ont tous été donnés, le client en est informé et est sorti du sas.
Un serveur de billetterie est mimé par le script *billetterie.sh.* Chaque serveur ne s'occupe que d'un seul client à la fois. Une fois élu, le client est pris en charge par un serveur. Il est notifié de son accès à la billetterie –ce qui le fait sortir de son attente -- et dispose d'un temps fixe (de 10s ici) pour choisir ses billets avant que le serveur ne passe au client suivant.

Conseil : dans cet exercice fil rouge, les questions ne se suivent pas forcément. **N'hésitez pas à passer.**

[2pts] Préambule

1)[0,5pt] Donnez **la** ligne de commande qui permet de créer un répertoire nommé *Billetterie* dans votre répertoire de connexion **quel que soit** votre positionnement dans le système de fichiers.

2)[0,5pt] Donnez **la** ligne de commande qui vous place dans le répertoire *Billetterie* **quel que soit** votre positionnement dans le système de fichiers.

3)[1pt] Un répertoire *Sources_CF* se trouve dans le répertoire parent du répertoire *Billetterie*. Sachant que vous êtes placés dans le répertoire *Billetterie*, donnez **la** ligne de commande qui déplace tous les scripts (fichiers qui ont été suffixés en **.sh)** du répertoire *Sources_CF* vers le répertoire de travail, en utilisant uniquement des chemins relatifs.

[6,5pts] Sas d'attente

Le script *sas_attente.sh* prend un unique paramètre correspondant au nombre de serveurs de billetterie devant accueillir les clients.

(a) [0,5pt] Vérifiez que le nombre d'arguments du script est correct. Si ce n'est pas le cas, affichez un message d'erreur et sortez avec un code d'erreur.

(b)[0,5pt] Initialisez une variable *nb_pass* à 10 fois le nombre de serveurs de billetterie indiqué.

(c)[1pt] Lancez le bon nombre de serveurs de billetteries, simulés par le script *billetterie.sh* lancé sans paramètre en arrière plan.

Toujours le script sas_attente.sh, passons à présent à la gestion des demandes d'accès des clients.
Les clients font leur demande d'accès au sas d'attente au travers d'un canal de communication. Répondant à cette requête, le sas d'attente répond via un canal de communication propre au client s'il le met en salle d'attente d'un serveur de billetterie libre ou si la demande est refusée. Cette réponse est conditionnée par le nombre d'accès aux serveurs de billetterie dont dispose le sas d'attente.

(d) [1pt] Mettez en place un canal de communication nommé *sas.pipe* et maintenez-le ouvert en lecture et écriture.

(e) [2,5pts] Écrivez une boucle qui, à chaque tour, lit le PID du client demandant un accès dans le canal de communication *sas.pipe*.
Si le nombre d'accès aux serveurs de billetterie -- stocké dans la variable *nb_pass* -- n'est pas épuisé,

- écrivez sur le canal de communication du client <PID>.pipe (ce canal qui a pour nom le numéro de pid du client suffixé par *.pipe* étant propre au client sera créé par le client lui-même dans le script *client.sh)* la chaîne de caractères *« En attente »*,
- décrémentez la variable *nb_pass*
- inscrivez le client en ajoutant son PID à la fin du fichier *billetterie.data.*

Sinon, écrivez sur le canal de communication du client « Complet ».

(f)[1pt] À ce stade, sans avoir de connaissance sur le comportement des scripts client.sh et billetterie.sh, quelles sont les données qui pourraient souffrir d'accès concurrents problématiques et devraient potentiellement être protégées ultérieurement ? Justifiez votre réponse.

[7pts] Serveur de billetterie

Un serveur de billetterie accueille en exclusivité un client placé en attente dans le fichier *billetterie.data*. Ainsi, dans une boucle infinie, dans *billetterie.sh*,

(g) [0,5pt] Faites une lecture de la première ligne du fichier billetterie.data dans une variable nommée *pid* (car elle correspond au PID du client à accueillir et permettra de dialoguer avec le processus *client.sh* correspondant aisément).

(h) [0,5pt] Si la chaîne de caractères lue est vide, attendez 3s avant de réitérer la lecture du fichier.

Sinon,

1. Retirez le client du fichier des clients en attente en suivant les étapes suivantes :

(i)[1,5pt] Commencez par déterminer le nombre de lignes du fichier billetterie.data et stockez le résultat dans une variable *nb_lignes*. Pour cela, construisez incrémentalement la commande sur votre brouillon suivant les instructions suivantes avant de la reporter sur votre copie.

- Comptez le nombre de lignes du fichier *billetterie.data* grâce à l'option -l de la commande *wc*.

- La commande produit un résultat dont le format est de la forme :

```
$ wc -l billetterie.data
2 billetterie.data
$
```

Sur une même ligne, il indique le nombre de lignes du fichier et rappelle le nom du fichier traité.
Ainsi, complétez votre commande en ne conservant que le premier champ du résultat produit par la commande précédente en prenant le soin d'utiliser le caractère « espace » comme délimiteur. La page de manuel de la commande *cut* vous permettrait de savoir que l'option *-f LISTE* (f pour *field)* permet de sélectionner le champ dont le numéro est listé et l'option -d, d'indiquer un nouveau délimiteur (celui par défaut est la tabulation).

- Stockez le résultat de la commande obtenue dans la variable *nb_lignes*.

(j)[1pt] Retirez le client en tête du fichier des clients en attente. Vous pouvez utiliser des solutions déjà vue en TP, ou bien utiliser le résultat de l'étape (i) en suivant les étapes suivantes :

- Utilisez la commande *tail* (dont la page de manuel est disponible en fin de sujet) pour ne garder que les nb_lignes -1 dernières lignes du fichier *billetterie.data ;*
- Écrivez le résultat de la commande dans un fichier *billetterie.data.new* ;
- Renommez le fichier *billetterie.data.new* en b*illetterie.data*.

2. Faites entrer le client dans la billetterie. Pour cela,

(k)[1pt] Notifiez à l'aide du signal USR1 le processus client qu'il entre dans la billetterie ;

(l) [0,5pt] Simulez le temps de choix des billets en laissant 10s au client ;

(m)[0,5pt]Notifiez le processus client à l'aide du signal USR2 de l'expiration de son créneau de choix de billets.

(n) [1,5 pts] En excluant l'interaction que pourrait avoir le script *client.sh*, l'exécution de sas_attente.sh engendre-t-il des problèmes de concurrence ? Répondez tout d'abord dans le scénario où sas_attente.sh est lancé avec 1 en paramètre, puis 5. Justifiez votre réponse, si nécessaire par un exemple de scénario problématique du point de vue de la concurrence.
En cas de problèmes de concurrence, protégez les sections critiques identifiées dans les scripts à l'aide des scripts P.sh et V.sh, vus en TP et disponibles dans le répertoire où vous travaillez. Faites précéder vos ajouts de la balise **(n)**.

[3,5pts] Client

Un client demande un accès auprès du sas d'attente de la billetterie des olympiades d'informatique en écrivant son PID dans le canal de communication ouvert par le sas d'attente, nommé sas.pipe. Il se met ensuite en attente de lecture d'une réponse sur son propre canal de communication ouvert au préalable. La réponse est soit « En attente » et le client se met en attente, soit « Complet » et le client arrête son traitement.

(o)[0,5pt] Ouvrez le canal de communication du client en le nommant suivant son numéro de PID suffixé par .pipe et maintenez le ouvert en lecture/écriture.

(p)[0,5pt] Écrivez dans le canal de communication sur sas d'attente le PID du client et lisez en retour la réponse du sas d'attente sur le canal de communication du client dans une variable *retour*.

Ensuite, le code de traitement de la réponse vous est donné. Il reste cependant à traiter la réception des notifications faites par le serveur de billetterie lorsque vient le tour du client de choisir ses billets et lorsque son temps de choix est achevé.

(q)[0,5pt] Écrivez le traitement de la notification liée à l'entrée dans la billetterie, lancée à la question (k), qui doit écrire sur la sortie standard du client le message « * Bienvenue à la billetterie * » et laisser retourner le client à son exécution initiale. Faites-le précéder de la balise (q) pour indiquer que vous répondez à la question (q) et prenez soin de protéger les caractères spéciaux.

(r)[0,5pt] Écrivez le traitement de la notification liée à la sortie dans la billetterie, lancée à la question (m), qui affiche le message « * Bravo pour vos achats *» et met un terme à l'activité du client. Faites-le précéder de la balise (r) pour indiquer que vous répondez à la question (r).

(s)[1,5pt] L'ensemble des scripts en main, leur exécution engendre-t-il des problèmes de concurrence ? Répondez tout d'abord dans le scénario où sas_attente.sh est lancé avec 1 en paramètre, puis 5. Justifiez votre réponse, si nécessaire par un exemple de scénario problématique du point de vue de la concurrence. En cas de problèmes de concurrence, protégez les sections critiques identifiées dans les scripts à l'aide des scripts P.sh et V.sh, vus en TP et disponibles dans lequel vous travaillez. Faites précéder vos ajouts de la balise **(s)**.

[1pt] Mise en exécution

(t)[1pt] Écrivez la suite de commandes qui permet de rendre exécutable pour l'utilisateur propriétaire l'ensemble des scripts venant d'être écrits et les lance dans un ordre correct. Vous pouvez utilisez plusieurs terminaux, numérotés suivant leur ordre d'utilisation. Ce scénario doit lancer 3 serveurs de billetterie et 4 clients.

Terminal 1 :

Terminal 2 :

Terminal 3 :

sas_attente.sh	**sas_attente.sh (suite)**
#! /bin/bash #(a) #(b) #(c) #(d)	#(e)

billetterie.sh

```
#! /bin/bash

while true ; do




  #(g)



  #(h)
  if [                    ] ; then



  else
     #(i)



     #(j)



     #(k)



     #(l)



     #(m)



fi




done
```

client.sh

```
#! /bin/bash




#(o)



#(p)



if [ "$retour" = "En attente" ]; then
    while  true ; do
     sleep 1
    done

else
    echo "Billetterie saturée."

fi
```

Annexe : page de manuel de la commande tail (extraits choisis)

NOM

tail - Afficher la dernière partie de fichiers

SYNOPSIS

tail [OPTION]... [FICHIER]...

Afficher les 10 dernières lignes de chaque FICHIER sur la sortie standard. Lorsqu'il y a plus d'un FICHIER, faire précéder chaque groupe de lignes d'un en-tête donnant le nom du fichier. L'entrée standard est lue quand FICHIER est omis ou quand FICHIER vaut « - ».

Les paramètres obligatoires pour les options de forme longue le sont aussi pour les options de forme courte.

-c, **--bytes**=K
afficher les K derniers octets ; vous pouvez aussi utiliser **-c** +K pour afficher les octets de chaque fichier à partir du Kième octet

-f, **--follow[=**{name|descriptor}]
afficher les dernières données ajoutées au fur et à mesure que le fichier s'accroît ; **-f**, **--follow**, et **--follow**=descriptor sont équivalents

-F identique à **--follow**=name **--retry**

-n, **--lines**=K
afficher les K dernières lignes, au lieu des 10 dernières ; ou utilisez **-n** +K pour afficher toutes les lignes à partir de la Kième.

--max-unchanged-stats=N
avec l'option **--follow**=name, réouvrir le FICHIER dont la taille n'a pas changé après N itérations (par défaut 5), afin de vérifier s'il a été détruit ou renommé (c'est habituellement le cas des fichiers journaux dont on effectue la rotation)

--pid=PID
avec **-f**, terminer après l'arrêt du processus PID

-q, **--quiet**, **--silent**
ne jamais afficher les en-têtes donnant les noms de fichiers

Si le premier caractère parmi K (nombre d'octets ou de lignes) est un « +», l'affichage commence à partir du Kième élément depuis le début de chaque fichier, sinon, les K derniers éléments du fichier sont affichés. K peut avoir un suffixe multiplicateur : b pour 512, kB pour 1000, K pour 1024, MB pour 1000*1000, M pour 1024*1024, GB pour 1000*1000*1000, G pour 1024*1024*1024, et ainsi de suite pour T, P, E, Z et Y.